clist와 keys

Nova De Hi

2014년 10월 20일

1 clist 샘플 (k_tst_5.tex)

```
1  \ExplSyntaxOn
2
3  \cs_new:Npn \test_map:n #1
4  {
5      맛있는~과일은~#1\par
6  }
7
8  \NewDocumentCommand \test { m m }
9  {
10     \clist_set:Nn \arg_clist { #1 }
11     \clist_put_left:Nn \arg_clist { #2 }
12     \clist_remove_duplicates:N \arg_clist
13
14     \clist_map_function:NN \arg_clist \test_map:n
15
16     \clist_count:N \arg_clist {~}items.
17
18     \clist_use:Nn \arg_clist {\과{~}}
19 }
20
21 \ExplSyntaxOff
```

line 1: ExplSyntaxOn

- \ExplSyntaxOn과 \ExplSyntaxOff 사이에 오는 코드는 expl3 문법을 따릅니다.
- @은 쓰지 않음. 그 대신 _와 :가 쓰입니다.

- 원칙적으로 모든 스페이스는 무시됨. 따라서 줄 끝에 %를 붙일 필요가 없습니다. 그 대신 스페이스를 살리기 위해서는 ~를 씀.
- 소스의 가독성을 높이기 위해서 띄어쓰기와 줄나눔을 빈번하게 사용합니다.

line 3: `\cs_new:Npn`

- expl3는 명령(command)을 모듈, 함수, 변수 세 수준으로 구분하여 정의합니다. `\cs_new:Npn`은 함수를 새로이 정의하라는 의미입니다. 여기서 `:Npn`은 인자지시자(argument specifier)라고 하는데 N은 한 개의 매크로 토큰, p는 parameter, n은 일반적인 (중괄호로 둘러싸인) 토큰(열)을 가리킵니다. 인자지시자는 이밖에도 `c, o, v, V, x, f, T, F, p, w` 등이 있습니다. 이 가운데 c는 특히 유용한데 예컨대 `\foo`라는 매크로를 `\csname foo\endcsname`이라고 쓰던 문법을 상기하시면 될 것입니다. 즉, 위의 line 3은 다음과 같이 써도 동일한 의미입니다.

  ```
  \cs_new:cpn {testmap:n} #1
  ```

- 새로 정의하는 함수명에 반드시 인자지시자가 있어야 합니다. 여기서는 한 개의 normal argument를 취할 것이기 때문에 `:n`을 붙입니다. (n은 `no manipulation`이라는 의미입니다.) 함수 이름이 같더라도 인자 지시자가 다르면 서로 다른 함수로 인식합니다. 즉 여기서 정의한 함수를 `\testmap`으로는 부를 수 없고 반드시 `\testmap:n`이라고 불러야 합니다.
- 한 개의 인자에 해당하는 `#1`을 써줍니다. 둘 이상이면 `#1 #2`와 같이 쓰면 됩니다. 이 부분이 p에 해당합니다.

line 4, 6: `{ ... }`

- 이 중괄호로 묶인 영역 전체가 앞서 `Npn`의 n에 해당합니다. 여기에 이 매크로의 의미(내용)를 기술합니다. 인자로 받은 것을 `#1`로 사용할 수 있는 것은 이전의 TEX에서와 동일합니다.
- `\cs_new:Npn`으로만 함수를 생성했고 `protected`를 지시하지 않았기 때문에 이 함수는 불리는 시점에서 확장됩니다. LATEX3는 매크로의 확장에 상당히 많은 주의를 기울여 만들어져 있기 때문에 예전과 같은 `\expandafter`의 남용을 거의 제거할 수 있습니다만, 이 문제는 이 글에서 더 다루지 않겠습니다.

line 8: `\NewDocumentCommand`

- 이 명령은 xparse 패키지에 정의되어 있고, 예전의 `\newcommand`와 역할이 비슷합니다. `\ProvideDocumentCommand`나 `\DeclareDocumentCommand`, `\RenewDocumentComma`와 같은 것도 있습니다.

- expl3로 쓰인 코드와 `\cs_new:Npn`으로 만들어진 함수는 본문, 즉 document 환경 내에서 바로 쓸 수 없습니다. 즉, expl3은 사용자 매크로를 만드는 것이 아닙니다. 사용자 인터페이스 명령은 xparse의 `\NewDocumentCommand`로 제공되어야 합니다.

- 이 예제에서 `\test`라는 이름의 사용자 인터페이스 명령을 정의하고 있습니다.

- `\NewDocumentCommand`의 최대 장점 중 하나는 인자의 유형을 미리 정의할 수 있다는 것입니다. 그것이 `{ m m }`이라고 지시된 부분입니다. 이것은 `m` 유형의 인자 두 개를 받는다는 의미입니다. 예전의 `\newcommand`가 `[2]`와 같은 방식으로 인자의 개수만을 지정해서 예컨대 옵션 인자를 정의하기 까다로웠던 것을 생각해보면 얼마나 편해진 건지 쉽게 알 수 있을 겁니다. 두 개 이상의 옵션 인자를 취하는 명령을 만들려면 온갖 방법을 동원해야 했던 지난날이 있다는 겁니다. 이제 `\@ifnextchar*`나 `\@ifstar` 따위 없이도 간단히 별표붙는 명령이나 옵션 인자를 취하는 명령을 정의할 수 있는 겁니다.

- 옵션 인자의 유형 중에서 꼭 알아두어야 하는 것은 다음과 같습니다.

  m mandatory. 일반적 표준적인 TEX 인자입니다. 중괄호로 묶여서 전달됩니다.

  o optional. 대괄호로 묶여서 전달되며 값을 가지지 않을 수 있는 선택인자입니다.

  O optional-default. `o`와 동일하지만 No Value일 때 가질 수 있는 기본값을 미리 설정합니다. `O{XXX}` 와 같은 식으로 지시합니다. 옵션 인자가 없으면 XXX를 전달하라는 뜻이겠지요.

  s star. 별표입니다. 별표가 있으면 `\IfBoolean`이 true가 됩니다.

  d delimited optional. 옵션인자와 같은데 대괄호가 아니라 주어지는 부호 문자로 전달하는 것입니다. 예를 들면 `d()` 와 같이 지시하면 괄호로 옵션인자를 전달할 수 있습니다.

  D delimited optional-default. `d`와 같은데 값이 없을 경우의 기본값을 설정해줄 수 있습니다. `D(){XXX}` 와 같은 식으로 지시합니다.

- 별표의 존재 여부는 \IfBooleanTF로 합니다. 예를 들어

```
\NewDocumentCommand \mytest { s o m }
{
    \IfBooleanTF {#1} { star } { nostar }
}
```

와 같이, 첫번째 인자가 s일 경우 별표가 있으면 \IfBooleanT 부분을 실행하게 할 수 있습니다. expl3의 (TF)는 (T), (F), (TF)로 쓰일 수 있고 각각 조건이 참일 때 실행할 코드, 거짓일 때 실행할 코드, 또는 참일 때와 거짓일 때 각각 실행할 코드를 그 뒤에 정의합니다.

- 옵션 인자가 주어지지 않았을 때는 \IfValueTF 또는 \IfNoValueTF로 이를 점검하여 실행할 코드를 설계합니다. 다음 코드를 보십시오.

```
\NewDocumentCommand \mytest { s o m }
{
    \IfBooleanTF {#1} { star } { nostar }
    \IfNoValueTF {#2} { no~ optional~ arguments }
                      { optional argument~ is ~#2}
}
```

line 10: clist datatype expl3의 clist 데이터타입은 쉼표로 구분된 리스트 데이터를 다루기 위한 것입니다. 이것을 간단한 데이터베이스 비슷하게 활용하거나 스택처럼 활용할 수도 있습니다.

clist를 정의하기 위해서 먼저 이름을 정해줍니다.

```
\clist_new:N \mytest_clist
\clist_set:Nn \mytest_clist {사과, 복숭아, 배, 참깨}
```

이제 \mytest_clist에는 네 개의 아이템이 저장되었습니다.

예제 제10행은 첫번째 인자를 받아들여서 \arg_clist에 저장하라는 의미입니다.

line 11: 아이템 추가 아이템은 현재의 리스트 앞쪽 또는 뒷쪽에 삽입할 수 있습니다. 각각 \clist_put_left:Nn, \clist_put_right:Nn입니다.

예제 제11행은 두번째 인자의 내용을 \arg_list의 앞쪽에 넣으라는 의미가 됩니다. 따라서

```
\test{사과, 복숭아, 배, 참깨}{포도, 복숭아, 사과}
```

와 같이 본문에서 쓰면 \arg_list에

포도, 복숭아, 사과, 사과, 복숭아, 배, 참께

라는 일곱 개의 아이템이 들어가게 되겠습니다.

line 12: 중복 아이템 제거 사과라는 아이템이 중복되어 있습니다. 이와 같이 동일한 아이템이 둘 이상 있을 때 그것을 제거하기 위해 \clist_remove_duplicates:N을 썼습니다.

이외에도 clist의 데이터를 조작하는 수많은 명령이 정의되어 있습니다.

line 14: map function clist의 각 아이템을 특정한 함수에 대하여 하나씩 실행시키는 것입니다. 이를 위해 우리는 \test_map:n 함수를 미리 만들어 두었습니다. \arg_clist를 이 함수에 대해서 매핑하면 아이템 개수만큼 하나씩 차례로 실행되는 것을 볼 수 있습니다.

```
22  \cs_new:Npn \testmap:n #1
23  {
24      내가~좋아하는~과일은~#1\par
25  }
26  \NewDocumentCommand \test { m }
27  {
28      \clist_set:Nn \arg_clist {#1}
29      \clist_map_function:NN \arg_clist \testmap:n
30  }
31  ...
32  \test{사과, 배, 복숭아}
```

내가 좋아하는 과일은 사과
내가 좋아하는 과일은 배
내가 좋아하는 과일은 복숭아

line 21, 23: clist 표현 21행의 \clist_count:N은 현재 리스트에 저장된 아이템의 개수를 보여줍니다.

23행의 \clist_use:Nn은 리스트의 아이템을 나열하게 하는 명령인데, 인자 n은 아이템 사이에 넣을 부호(쉼표라든가)를 지정합니다. 이의 확장판으로 \clist_use:Nnnn이 있는데요, 여기 세 개의 n인자는 각각 아이템이 두 개일 때의 연결부호, 아이템이 셋 이상일 때의 연결부호, 아이템이 셋 이상일 때의 마지막 연결부호를 지정합니다. 예를 들어

\clist_use:Nnnn \arg_list {~and~} {,~} {,~and~}

와 같이 지정하면

사과, 복숭아, 배, and 포도

와 같이 나타납니다. 만약 아이템이 두 개라면

사과 and 복숭아

가 되겠지요.

이밖에도 clist를 다루는 방법이 많습니다. 매뉴얼을 보고 연습해보시기 바랍니다.

2 keys

우리가 흔히 쓰는 `\includegraphics` 명령에 보면

```
\includegraphics[width=3cm,height=5cm]{fig}
```

와 같이 [*key*=*value*]와 같은 표현이 있습니다. 이것을 key-value 표현이라고 하고, keyval 패키지나 xkeyval 패키지로 이와 같은 표현을 구현해왔습니다. expl3에는 이 데이터 구조가 미리 정의되어 있어서 xkeyval같은 고생을 하지 않아도 되고, 특히 l3keys2e 패키지를 통하여 자신이 작성하는 패키지의 옵션으로 key-val 표현을 바로 쓸 수 있게 해주고 있기도 합니다.

여기서는 정말 단순한 예를 하나만 들고자 합니다.

```
33  \ExplSyntaxOn
34
35  \keys_define:nn { mytest }
36  {
37      color   .tl_set:N  = \l_mytestcolor_tl,
38      size    .dim_set:N = \l_mytestsize_dim
39  }
40
41  \NewDocumentCommand \test { o m }
42  {
43      \IfNoValueTF { #1 }
44      {
45          \keys_set:nn { mytest }
46          {
47              color = red,
48              size  = 10pt
49          }
50      }
51      {
```

```
52          \keys_set:nn { mytest } { #1 }
53      }
54      \textcolor{ \l_mytestcolor_tl }
55      {
56          \fontsize{ \l_mytestsize_dim }
57                    { \l_mytestsize_dim }
58          \selectfont #2
59      }
60  }
61
62  \ExplSyntaxOff
```

line 35: keys define

- `\keys_define:nn` 명령의 첫번째 인자는 *module*입니다. 하나의 유니크한 모듈 이름을 주어야 합니다.
- 두번째 인자는 *key*와 그 처리를 지정합니다. 이 부분이 주의가 좀 필요한데요, 여기서 쓸 수 있는 함수의 예를 들면 다음과 같습니다.
  - `.tl_set:N, .dim_set:N, .skip_set:N, .fp_set:N, .clist_set:N, .seq_set:N ...`
  - `.code:n, .meta:n, .default:n, ...`
- 예컨대 `color .tl_set:N = \mycolor_tl`,이라고 정의했다면, 이것은 *key*가 color라는 것이고 `[color=XXX]`와 같이 주어진 key-val 표현에서 XXX에 해당하는 부분을 `\mycolor_tl`이라는 token list로 넣으라는 의미입니다.
- 여기서는 간단히 색상 이름에 해당하는 tl, 그리고 크기에 해당하는 dim을 각각 color와 size라는 키로 설정하는 예를 보였습니다.

line 41: key set

- key-value expression은 보통 사용자 명령의 옵션 인자로 주게 됩니다. 그래서 `o`가 지시되었고, 이 `#1`에 key-val 리스트가 오게 될 것입니다.
- 만약 이 옵션 인자가 비어 있으면 어떻게 처리할 것인가가 44에서 50행까지의 내용입니다. 즉 `[color=red,size=10pt]`라는 것을 디폴트로 설정하고 이 값으로 mytest module의 keys를 setting하라는 것이 `\keys_set:nn` 명령의 의미입니다.

- 51–53행은 옵션 인자로 받아들인 key-val expression을 그대로 `\keys_set:nn` 하는 것이네요.
- 아무튼 여기까지 하면, mytest 모듈의 color와 size 키가 처리되어 `\l_mytestcolor_tl`과 `\l_mytestsize_dim`이라는 매크로가 정의되게 됩니다. 이것을 이용하여 `\textcolor{색상}{크기} #2`로 식자하도록 정의한 부분이 54–59행의 내용입니다.
- 즉, 이 명령 `\test[color=blue,size=20pt]{My text}`는 blue 색상과 20pt 크기로 My text라는 텍스트를 찍습니다.

```
63  \test[color=blue,size=20pt]{테스트입니다}
```

테스트입니다